CLAIMS

1. A method of optimizing scalability in a multiprocessor data server having N processors, wherein N is an integer greater than or equal to 2, the method comprising:

implementing N NICs (Network Interface Cards), a first one of the N NICs being dedicated to receiving an incoming data stream;

binding an interrupt from the first one of the N NICs to a first one of the N processors;

binding an interrupt for an nth NIC to an nth processor, wherein $0 < n < = N$; and

binding a DPC (Deferred Procedure Call) for the nth NIC to the nth processor.

2. The method of claim 1, further comprising tightly coupling M client connections to the nth processor via the nth NIC, wherein M is a positive integer.

3. The method of claim 1, further comprising binding P server threads to specific ones of second through N processors, wherein P is a positive integer.

4. The method of claim 2, further comprising binding P server threads to specific ones of second through N processors, wherein P is a positive integer.

5. The method of claim 1, further comprising:

defining L1 (Level 1) and L2 (Level 2) caches for each of the N processors;

storing instructions and temporal data in L2 caches of the N processors; and

storing non-temporal data in L1 caches of the N processors, bypassing the L2 caches.

6. The method of claim 2, further comprising:

defining L1 (Level 1) and L2 (Level 2) caches for each of the N processors;

storing instructions and temporal data in L2 caches of the N processors; and

storing non-temporal data in L1 caches of the N processors, bypassing the L2 caches.

7. The method of claim 3, further comprising:

defining L1 (Level 1) and L2 (Level 2) caches for each of the N processors;

storing instructions and temporal data in L2 caches of the N processors; and

storing non-temporal data in L1 caches of the N processors, bypassing the L2 caches.

8. The method of claim 4, further comprising:

defining L1 (Level 1) and L2 (Level 2) caches for each of the N processors;

storing instructions and temporal data in L2 caches of the N processors; and

storing non-temporal data in L1 caches of the N processors, bypassing the L2 caches.

9. A method of optimizing scalability in a multiprocessor data server having N processors, wherein N is an integer greater than or equal to 2, the method comprising:

implementing N NICs (Network Interface Cards); and

tightly coupling M client connections to the nth processor via the nth NIC, wherein M is a positive integer and wherein 0 < n < = N.

10. The method of claim 9, further comprising binding P server threads to specific ones of second through N processors, wherein P is a positive integer.

11. The method of claim 10, further comprising:

defining L1 (Level 1) and L2 (Level 2) caches for each of the N processors;

storing instructions and temporal data in L2 caches of the N processors; and

storing non-temporal data in L1 caches of the N processors, bypassing the L2 caches.

12. The method of claim 9, further comprising:

defining L1 (Level 1) and L2 (Level 2) caches for each of the N processors;

storing instructions and temporal data in L2 caches of the N processors; and

storing non-temporal data in L1 caches of the N processors, bypassing the L2 caches.

13. A method of optimizing scalability in a multiprocessor data server having N processors, wherein N is an integer greater than or equal to 2, the method comprising:

implementing N NICs (Network Interface Cards); and

binding P server threads to specific ones of a second through N processors.

14. The method of claim 13, further comprising:

defining L1 (Level 1) and L2 (Level 2) caches for each of the N processors;

storing instructions and temporal data in L2 caches of the N processors; and

storing non-temporal data in L1 caches of the N processors, bypassing the L2 caches.

15. A method of optimizing scalability in a multiprocessor data server having N processors, wherein N is an integer greater than or equal to 2, the method comprising:

implementing L1 (Level 1) and L2 (Level 2) caches for each of the N processors;

storing instructions and temporal data in L2 caches of the N processors; and

storing non-temporal data in L1 caches of the N processors, bypassing the L2 caches.

16. The method of claim 5, further comprising improving L1 cache efficiency by increasing a time quantum allotted to server threads which process streaming data buffers.

17. The method of claim 6, further comprising improving L1 cache efficiency by increasing a time quantum allotted to server threads which process streaming data buffers.

18. The method of claim 7, further comprising improving L1 cache efficiency by increasing a time quantum allotted to server threads which process streaming data buffers.

19. The method of claim 8, further comprising improving L1 cache efficiency by increasing a time quantum allotted to server threads which process streaming data buffers.

20. The method of claim 11, further comprising improving L1 cache efficiency by increasing a time quantum allotted to server threads which process streaming data buffers.

21. The method of claim 14, further comprising improving L1 cache efficiency by increasing a time quantum allotted to server threads which process streaming data buffers.

22. The method of claim 15, further comprising improving L1 cache efficiency by increasing a time quantum allotted to server threads which process streaming data buffers.

23. A multiprocessor data server comprising:

N processors, wherein N is an integer greater than or equal to 2;

N NICs (Network Interface Cards), a first one of said N NICs being dedicated to receiving an incoming data stream;

wherein an interrupt from the first one of said N NICs is bound to a first one of said N processors; and wherein an interrupt for an nth NIC is bound to an nth processor, $0 < n < = N$; and

wherein a DPC (Deferred Procedure Call) for said nth NIC is bound to said nth processor.

24. The multiprocessor data server of claim 23, further comprising M client connections, wherein said M client connections are tightly coupled to said nth processor via said nth NIC, M being a positive integer.

25. The multiprocessor data server of claim 23, further comprising P server threads, wherein said P server threads are bound to specific ones of a second through N processors.

26. The multiprocessor data server of claim 24, further comprising P server threads, wherein said P server threads are bound to specific ones of a second through N processors.

27. The multiprocessor data server of claim 23, further comprising L1 (Level 1) and L2 (Level 2) caches for each of said N processors, wherein instructions and temporal data are stored in said L2 caches of said N processors, and wherein non-temporal data is stored in L1 caches of said N processors, bypassing the L2 caches.

28. The multiprocessor data server of claim 24, further comprising L1 (Level 1) and L2 (Level 2) caches for each of said N processors, wherein instructions and temporal data

are stored in said L2 caches of said N processors, and wherein non-temporal data is stored in L1 caches caches of said N processors, bypassing the L2 caches.

29. The multiprocessor data server of claim 25, further comprising L1 (Level 1) and L2 (Level 2) caches for each of said N processors, wherein instructions and temporal data are stored in said L2 caches of said N processors, and wherein non-temporal data is stored in L1 caches caches of said N processors, bypassing the L2 caches.

30. The multiprocessor data server of claim 26, further comprising L1 (Level 1) and L2 (Level 2) caches for each of said N processors, wherein instructions and temporal data are stored in said L2 caches of said N processors, and wherein non-temporal data is stored in L1 caches caches of said N processors, bypassing the L2 caches.

31. A program storage device, readable by a machine, embodying a program of instructions executable by the machine to perform a method of optimizing scalability in a multiprocessor data server having N processors, wherein N is an integer greater than or equal to 2, the method comprising:

implementing N NICs (Network Interface Cards), a first one of the N NICs being dedicated to receiving an incoming data stream;

binding an interrupt from the first one of the N NICs to a first one of the N processors;

binding an interrupt for an nth NIC to an nth processor, wherein $0 < n < = N$; and

binding a DPC (Deferred Procedure Call) for the nth NIC to the nth processor.

32. The program storage device of claim 31, the method further comprising tightly coupling M client connections to the nth processor via the nth NIC, wherein M is a positive integer.

33. The program storage device of claim 31, the method further comprising binding P server threads to specific ones of second through N processors, wherein P is a positive integer.

34. The program storage device of claim 32, the method further comprising binding P server threads to specific ones of second through N processors, wherein P is a positive integer.

35. The program storage device of claim 31, the method further comprising:

defining L1 (Level 1) and L2 (Level 2) caches for each of the N processors;

storing instructions and temporal data in L2 caches of the N processors; and

storing non-temporal data in L1 caches of the N processors, bypassing the L2 caches.

36. The program storage device of claim 32, the method further comprising:

defining L1 (Level 1) and L2 (Level 2) caches for each of the N processors;

storing instructions and temporal data in L2 caches of the N processors; and

storing non-temporal data in L1 caches of the N processors, bypassing the L2 caches.

37. The program storage device of claim 33, the method further comprising:

defining L1 (Level 1) and L2 (Level 2) caches for each of the N processors;

storing instructions and temporal data in L2 caches of the N processors; and

storing non-temporal data in L1 caches of the N processors, bypassing the L2 caches.

38. The program storage device of claim 34, the method further comprising:

defining L1 (Level 1) and L2 (Level 2) caches for each of the N processors;

storing instructions and temporal data in L2 caches of the N processors; and

storing non-temporal data in L1 caches of the N processors, bypassing the L2 caches.